MODULE-3

NON-FUNCTIONAL TESTING

Nonfunctional testing

- **Non-functional Testing** is a type of software testing that is performed to verify the non-functional requirements of the application.
- It verifies whether the behavior of the system is as per the requirement or not.

Non-Functional Testing Types and Techniques

- **Performance Testing**:
- Performance testing is a type of software testing that focuses on evaluating the performance and scalability of a system or application.
- Performance testing aims to identify bottlenecks, measure system performance under various loads and conditions, and ensure that the system can handle the expected number of users or transactions.

- **Load Testing**: Load testing determines the behavior of the application when multiple users use it at the same time.
- It is the response of the system measured under varying load conditions.
- **Security Testing**: Security testing is important to check and sure that applications and systems are protected from various threats.
- There are several types of security testing, each targeting specific vulnerabilities and aspects of security.

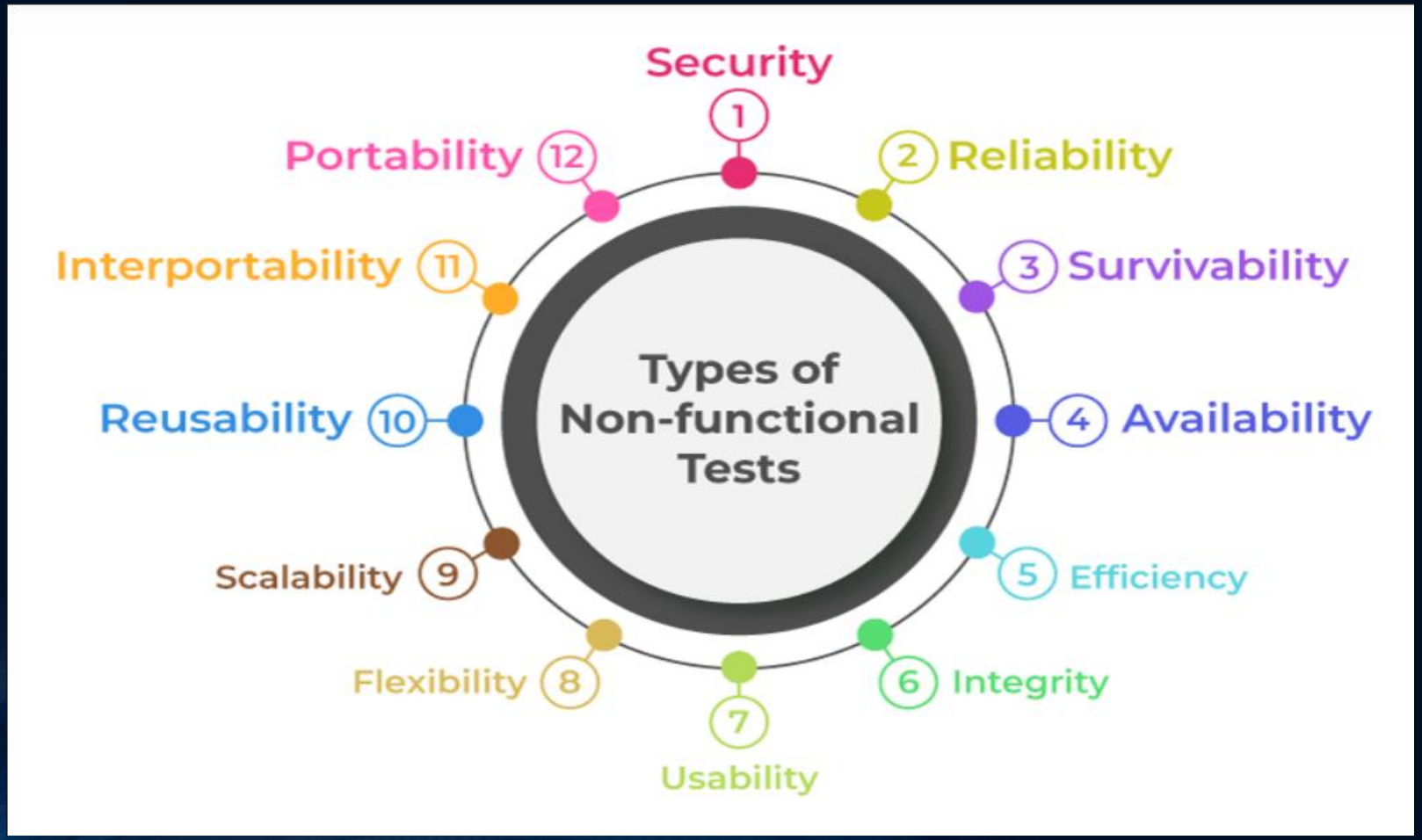
Objectives of Non-functional Testing

Objectives of Non-functional Testing

- **Increased usability:** To increase usability, efficiency, maintainability, and portability of the product.
- **Reduction in production risk:** To help in the reduction of production risk related to non-functional aspects of the product.
- **Cost Reduction:** To help in the reduction of costs related to non-functional aspects of the product.
- **Optimize installation:** To optimize the installation, execution, and monitoring way of the product.
- **Collect metrics:** To collect and produce measurements and metrics for internal research and development.
- **Enhance knowledge of product:** To improve and enhance knowledge of the product behavior and technologies in use.

Non-functional Testing Parameters

- The parameters are often used to ensure a system's performance, reliability, usability, and other quality attributes.

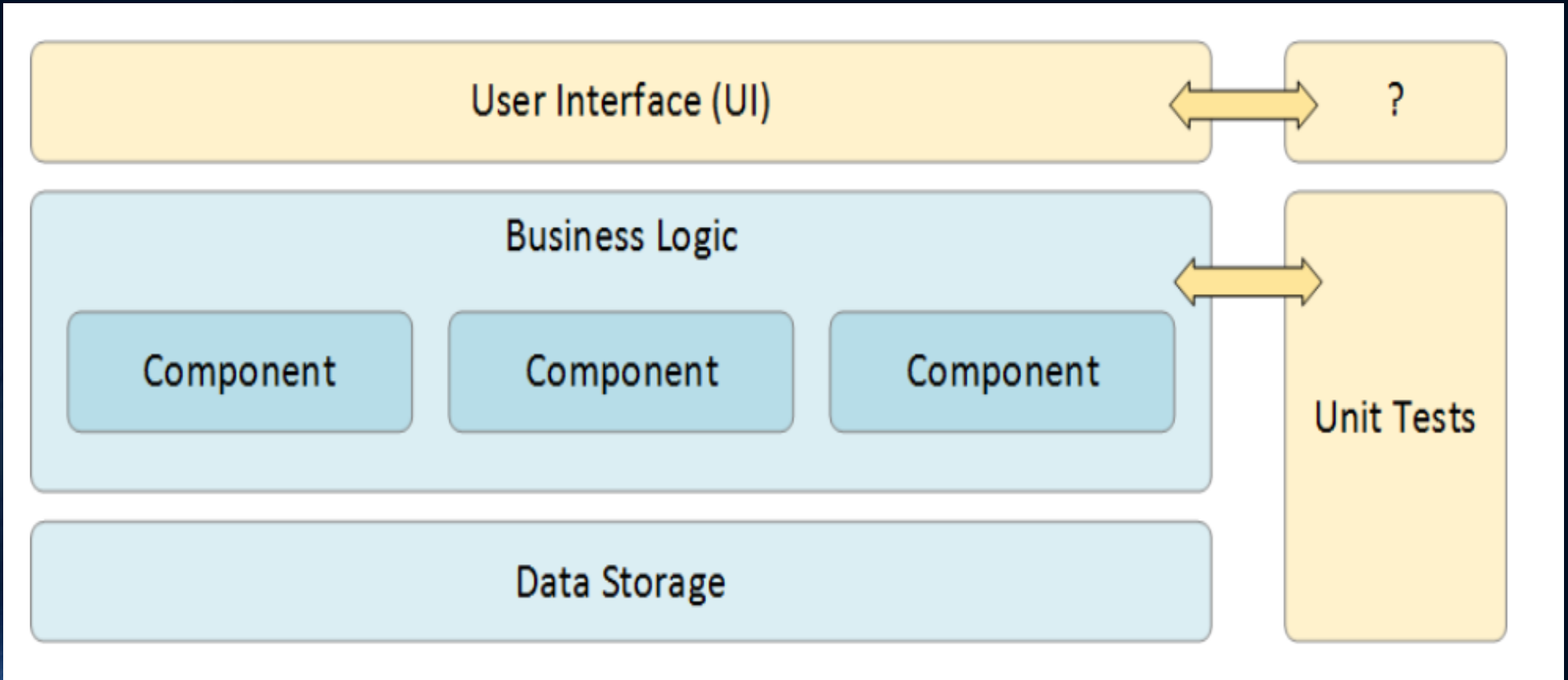


- **Security:** This parameter is tested during Security testing. This parameter defines how the system is secure against sudden attacks from internal and external sources.
- **Reliability:** This parameter is tested during Reliability testing. This defines the extent to which the system performs its intended functions without failure.
- **Survivability:** This parameter is tested during Recovery testing. This parameter checks that the software system is able to recover itself in the case of failure and continuously performs the specified function without any failure.
- **Availability:** This is tested during Stability testing. Availability here means the availability percentage of the software system to the original service level agreement. It means the degree to which the user can rely on the software during its operation.
- **Efficiency:** This parameter means the extent to which the software system can handle the quantity and response time.
- **Integrity:** This parameter measures how high the source code quality is when it is passed on to the QA.

- **Usability:** This is tested in usability testing. This parameter means how easily usable the system is from the user's perspective.
- **Flexibility:** This parameter means how well the system can respond to uncertainty in a way that allows it to function normally.
- **Scalability:** This parameter is tested during scalability testing. This parameter measures the degree to which the application can scale up or scale out its processing capacity to meet an increase in demand.
- **Reusability:** This means how many existing assets can be reused in some form within the software product development process or in another application.
- **Interoperability:** This parameter is tested during the Interoperability testing. This checks that the application interfaces properly with its components or other application or software.
- **Portability:** This parameter checks the ease with which the software can be moved from one environment to another.

- Advantages:
- **Improved performance**
- **Less time-consuming**
- **Improves user experience**
- **More secure product**
- Dis-advantages:
- **Non-functional tests are performed repeatedly**
- **Expensive in case of software update**

GUI Testing


User Interface (UI)
?
Business Logic
Component
Component
Component
Unit Tests
Data Storage

GUI TESTING

GUI testing refers to the validating UI functions or features of an application that are visible to the users, and they should comply with business requirements.

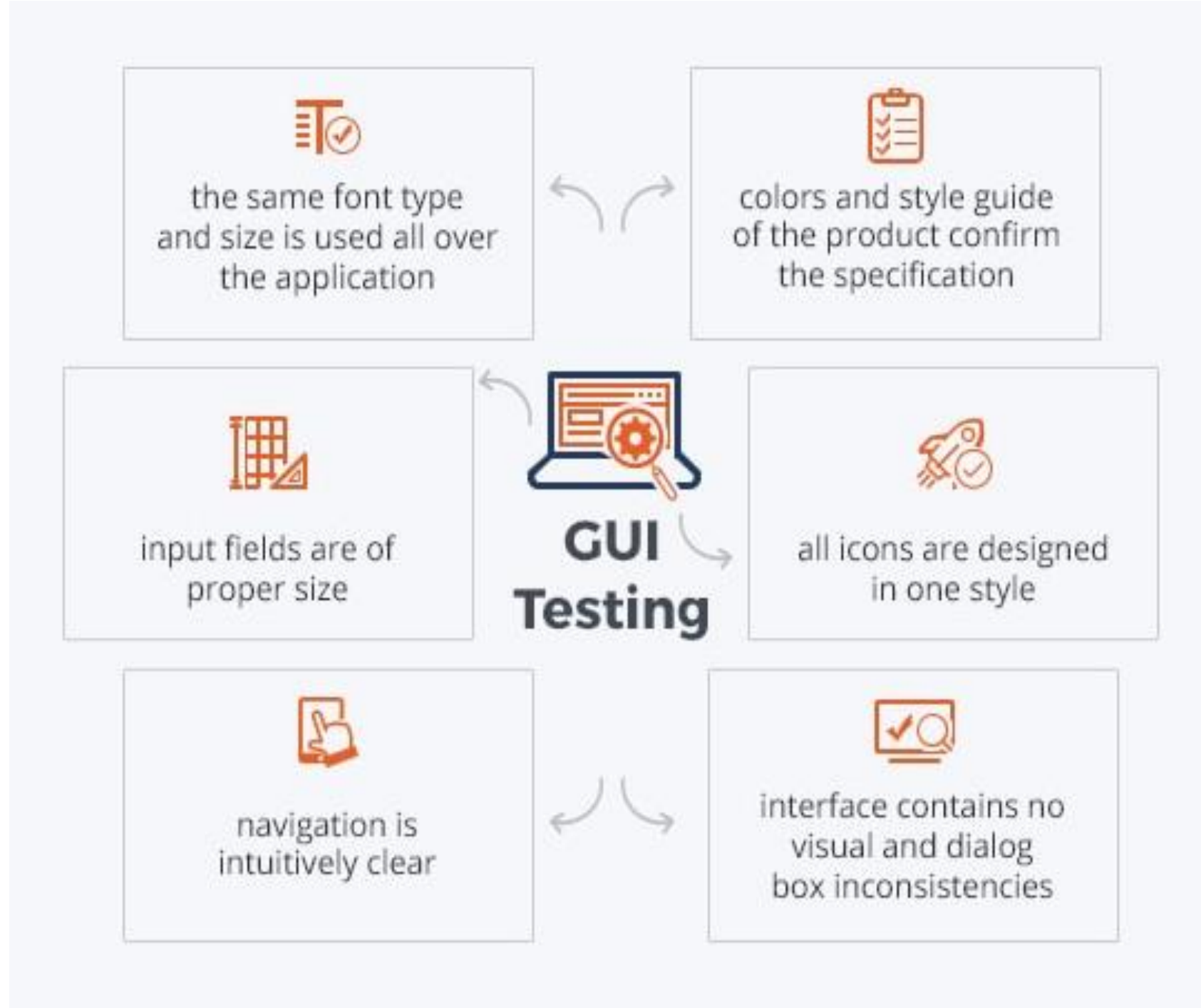
GUI testing is also known 'User Interface testing’.

TYPES OF GUI TESTING

1. Analog Recording

2. Object based Recording

PURPOSE OF GUI TESTING



ANALOG RECORDING

This is often what people associate with GUI testing tools. With analog recording, the testing tool basically captures specific mouse clicks, keyboard presses and other user actions and then simply stores them in a file for playback. For example, it might record that a user left-clicked at position X = 500 pixels, Y = 400 pixels or typed the word “Search” in a box and pressed the [ENTER] key on their keyboard.

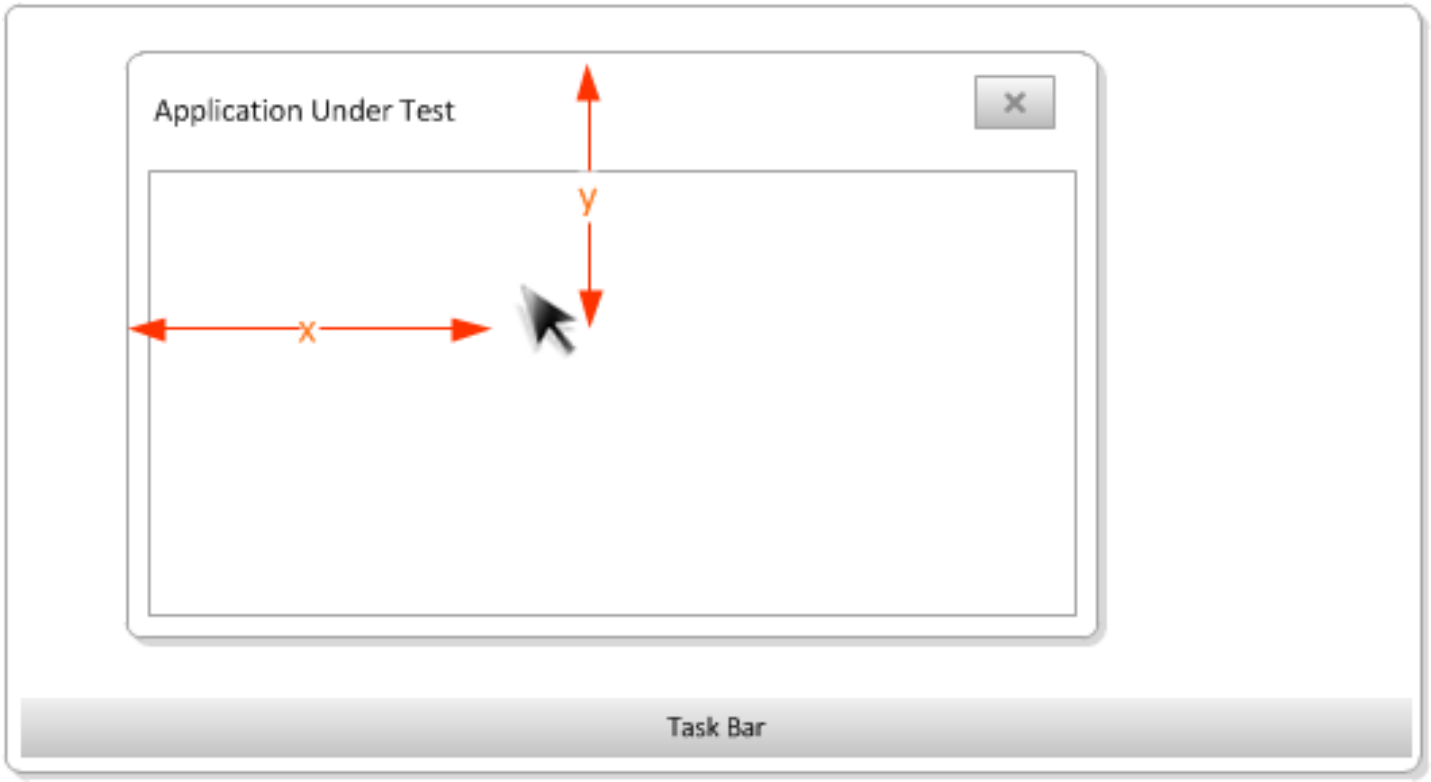
Types of Analog Recording:

1. Relative Analog Recording

2. Absolute Analog Recording

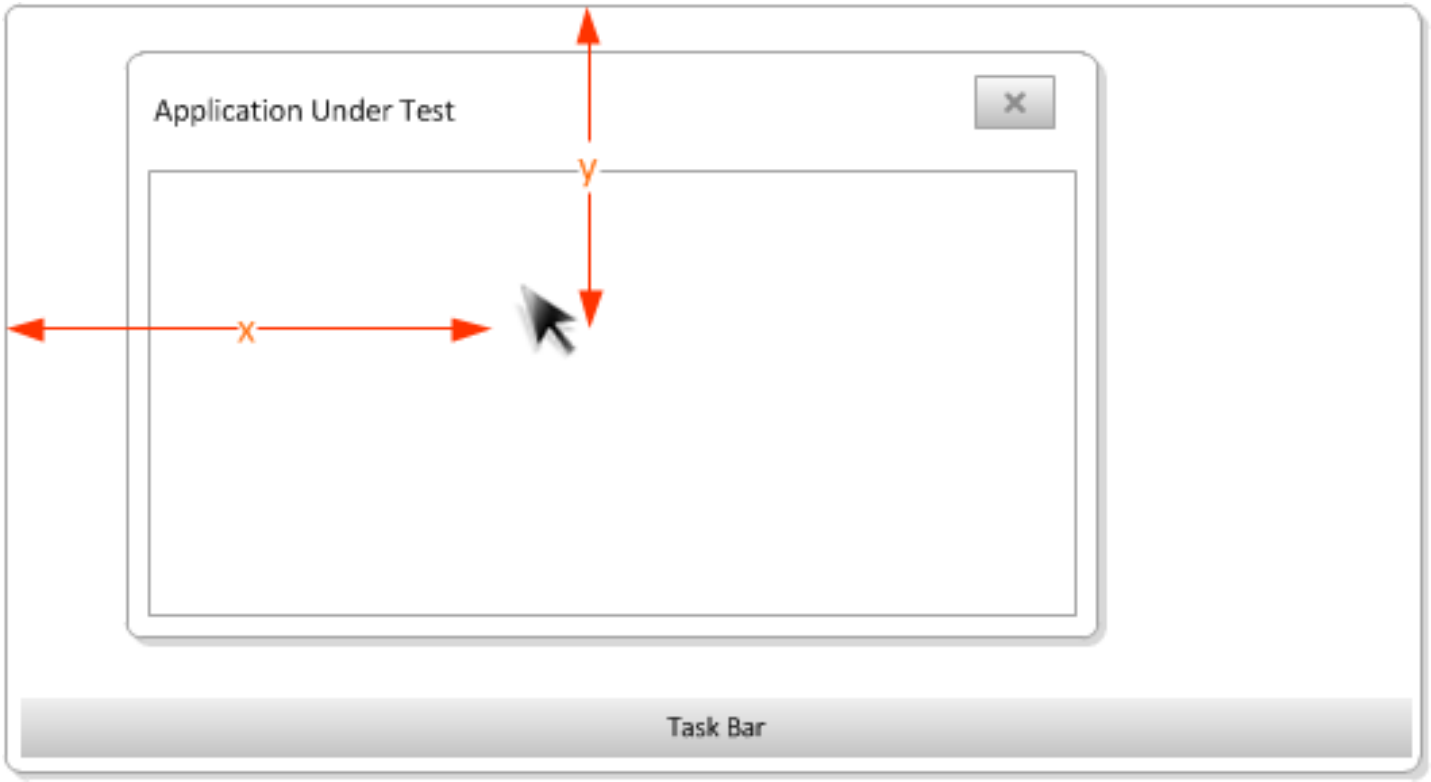
Relative Analog Recording

Relative analog recording is one type of analog recording where the positions of interaction events are recorded in relation to the top-left corner of the application's window:



Absolute Analog Recording

Absolute analog recording is the other main type of analog recording. Here, the positions of events are recorded relative to the top-left corner of the system screen.



Benefits of Analog Recording

- It works with virtually all applications regardless of the technology or platform used. Nothing is required of the application developer to make it testable.

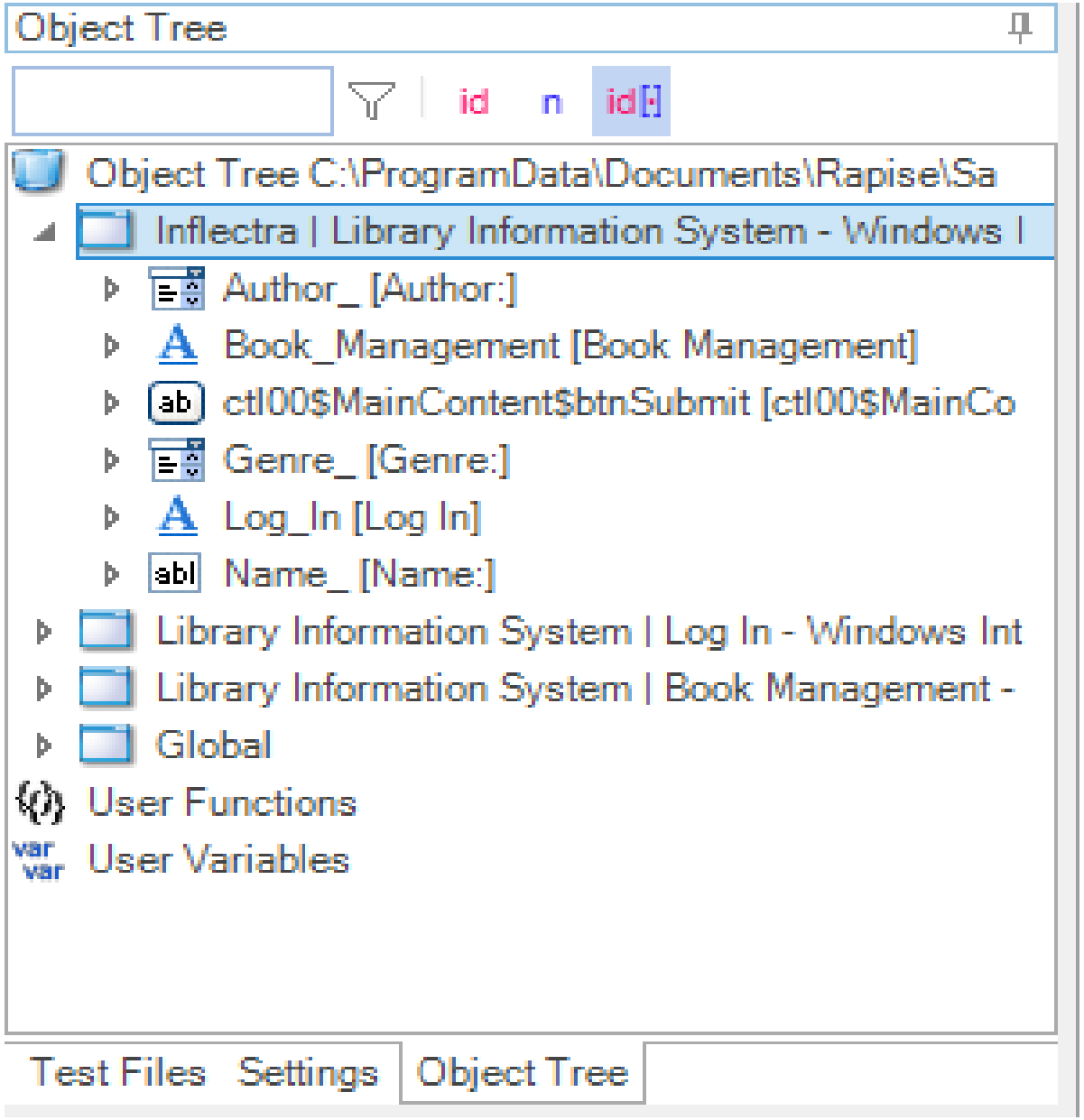
- It is quick to write such tests and as long as the GUI is stable and you can guarantee the screen resolution and window position, it can be a good way to test older applications that are not under active development

Drawbacks to Analog Recording

- The tests are sensitive to changes in the screen resolution, object position the size of windows, and scrollbars. That means you need to spend extra effort making sure you can standardize such factors.
- The tests are not intelligent; they don't look for specific objects (e.g. the Submit button) but are just a series of recorded gestures. So, if you change the UI (e.g. move a button or change a button to a hyperlink) in any way, the tests will need be rewritten. For this reason, analog tests are referred to as "brittle".
- When analog tests perform an action, there is very limited feedback since the testing tool is just clicking at a coordinate or sending a keypress – the tool does not understand if the application worked correctly, making human validation essential.

OBJECT BASED RECORDING

When you use object-based recording, the testing tool can connect programmatically to the application being tested and “see” each of the individual user interface components (a button, a text box, a hyperlink) as separate entities and is able to perform operations (click, enter text) and read the state (is it enabled, what is the label text, what is the current value) reliably regardless of where that object is on the screen.



Benefits of Object based Recording

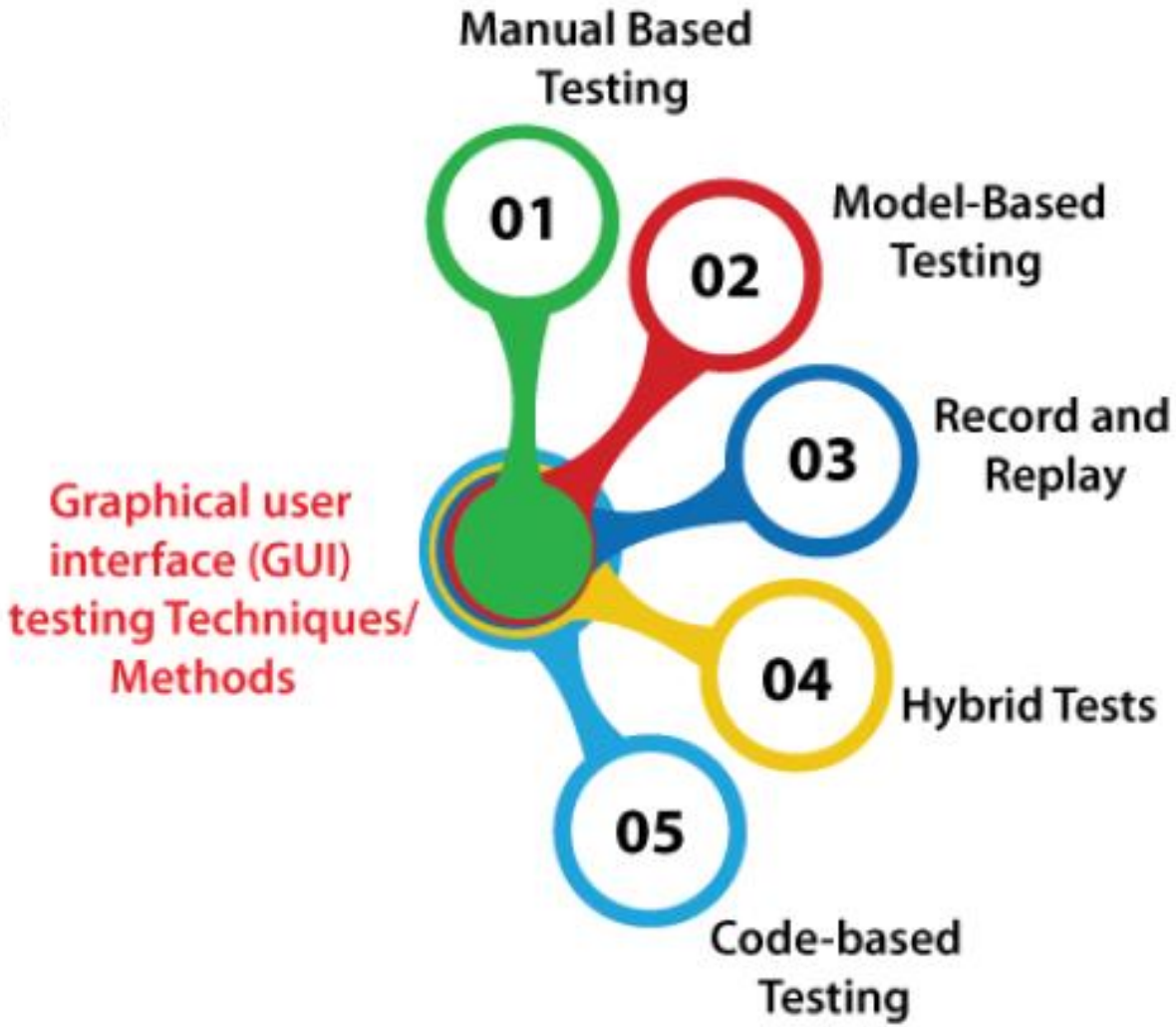
- The test is more robust and does not reply on the UI objects being in a certain position on the screen or being in the top-most window, etc.
- The test will give immediate feedback when it fails, for example that a button could not be clicked or that a verification test on a label did not match the expected text.
- The tests will be less “brittle” and require less rework as the application changes. This is especially important for a system under active development that is undergoing more UX changes. For older, legacy applications this may be less important.

Drawbacks of Object based Recording

The testing tool needs to have specific support for each of the technologies being used in the application. For example, if you have a web page that contains a Java applet and a Flash application embedded, you will need to have a testing tool that understands all three technologies (Java UI Toolkit, Flex Runtime and the HTML DOM).

- For some technologies (e.g. Flex, Flash) the developers need to add instrumentation to their code so that the testing tools able to “see” the UI objects. If you don’t have access to the source code of the application, then it’s not possible to test it.
- Writing the tests may take more skill than simply clicking and pointing, you need to be able to use Spy and inspection tools to navigate the object hierarchy and interact with the right UI elements.

GUI TESTING METHODOLOGIES

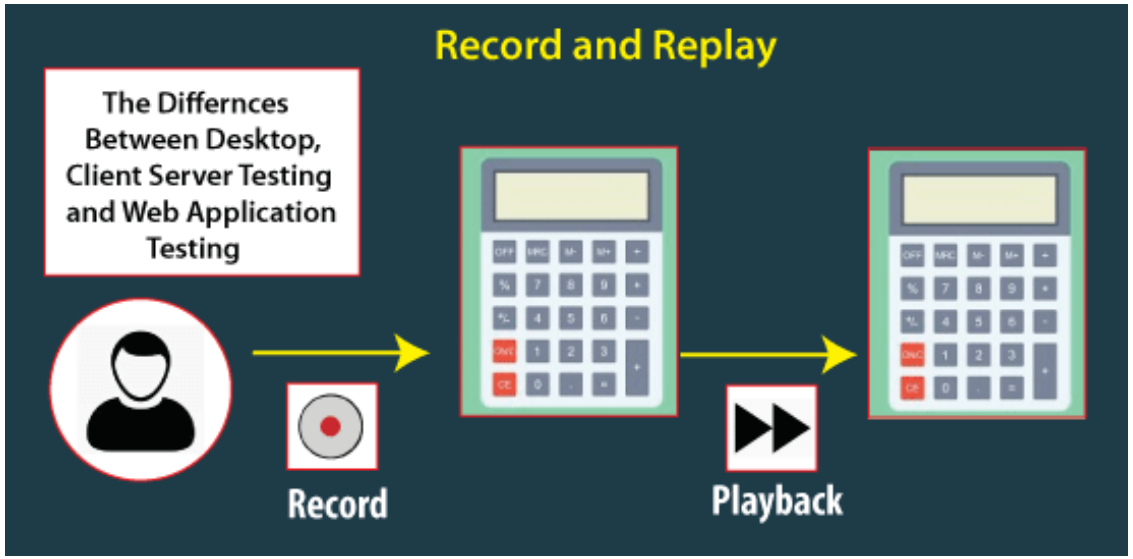


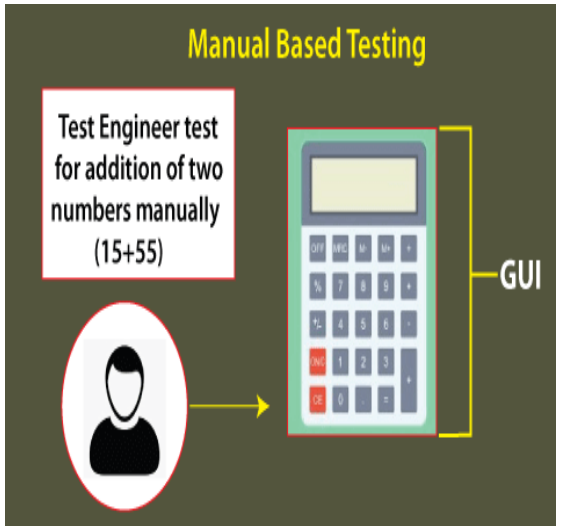
Manual Based Testing, Model Based testing, Record and Playback

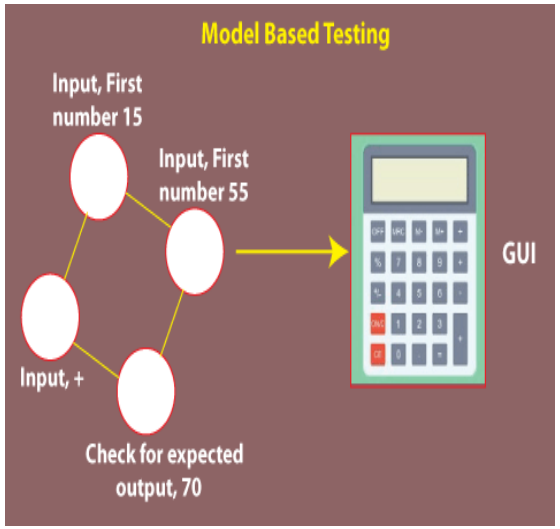
The graphical projects are tested **manually** by the test engineer following the requirements specified in the **BRS (Business Requirements Specification**) document.

Model-based Testing is a visual narrative of System performance, which helps us to understand and predict the system performance or activity.

We can perform the GUI testing with the help of Automation tools, which can be completed in two types. Throughout the **record** part, the test steps are encapsulated by the automation tool. And in the **playback**, these recorded test steps are implemented on the application under test. **For example: QTP, Selenium IDE**







4. Hybrid Tests

The hybrid tests are the different approach in order to perform GUI testing at the current time. It is a beneficial technique for non-technical background users to develop a test case by recording their sessions. And after that, the user who is familiar with coding can further control these recorded tests technically.

And the person who have the coding knowledge can further manipulate these recorded tests to modify them for more complex situations.

5. Code-based Testing

Another method of performing Graphical user interface testing is Code-based testing. In order to develop test cases by using the code the GUI testing provides some GUI testing tools. To discover more difficult test scenarios, we can use the code-based testing approach.

Subsequently, their code, test cases can be design in source control along with the application's code.

Commonly used GUI testing tools

- **Ranorex Studio**
- **Eggplant**
- **Squish**
- **AutoIT**
- **RIATest**
- **QTP**
- **TestComplete**
- **Selenium**

DOMAIN BASED TESTING

It is a software testing technique where minimum numbers of inputs are used to access appropriate output of a system, to ensure the system does not accept invalid input values. The system is expected to give required outputs blocking the invalid inputs.

Structure of Domain Testing :
The process is quite similar everywhere when it comes to building the strategy, where the following step-by-structure is used that suits most of the scenarios:

- Think what can go wrong.
- Find a solution to handle each case.
- Pick several points to test each error.
- Take one test point to examine adjacent domains
- Then start running the test
- Check if the boundaries are faulty and Inspect boundaries of all domains

Domain tester should have basic domain knowledge. It is important because:

- **Online banking –**
  A tester must have to be an expert in online banking activities like login, bill payment, and transfers.
- **Retail domains –**
  To successfully run a domain test, the tester has to recognize how things work flow at different levels. Some examples of retail domains are warehouse management, in-store solutions, etc.
- **Healthcare –**
  A tester with a proper understanding of domain knowledge should handle a healthcare system. It is a huge risk to someone's life when someone with zero knowledge handles the system.

Advantages of Domain Testing

- Identifies edge cases that can cause issues in a software system.
- Efficient use of testing resources.
- Increases test coverage by testing a wide range of values within a domain.
- Improves software quality by identifying and fixing issues in edge cases.
- Cost-effective testing technique that requires fewer resources compared to other testing techniques.

Disadvantages of Domain Testing

- Limited scope, only focuses on a specific range or domain.
- A false sense of security, may not identify all issues outside the domain.
- Difficult to determine domain boundaries.
- May not catch complex issues that require testing of multiple variables or interactions between different parts of the system.
- May not be suitable for all types of systems, particularly those that have complex input and output parameters or that require more advanced testing techniques.